

计算机学院课程

计算机组成原理

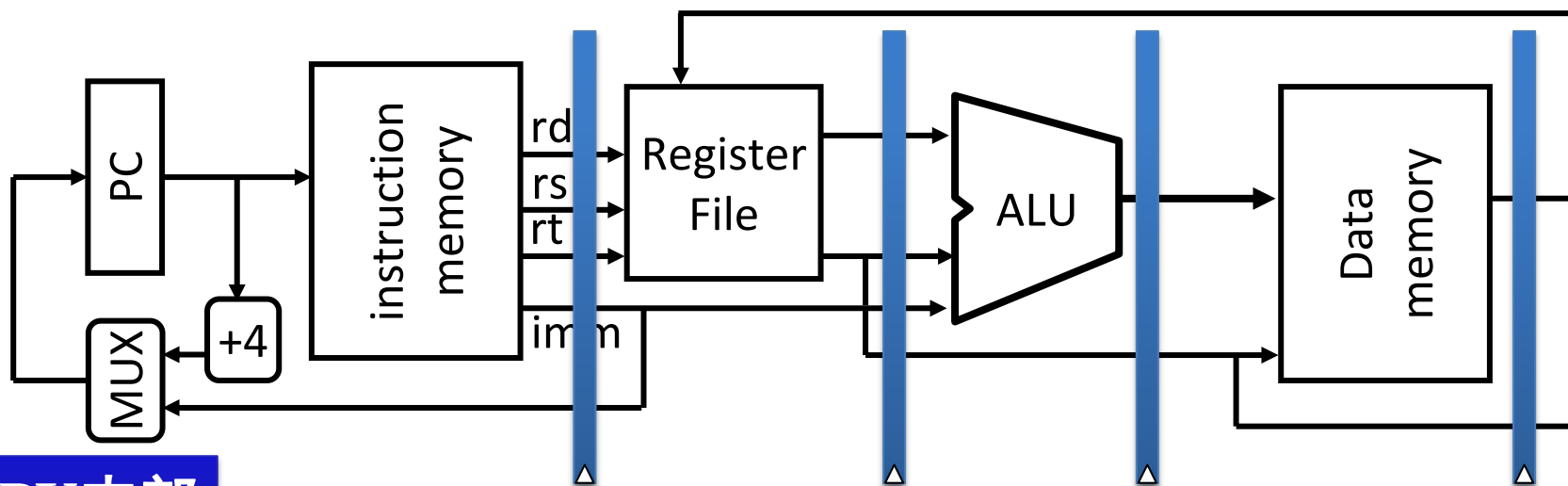
支持I/O

高小鹏

北京航空航天大学计算机学院
系统结构研究所

数据通路：增加信号

- 增加必要的地址、数据



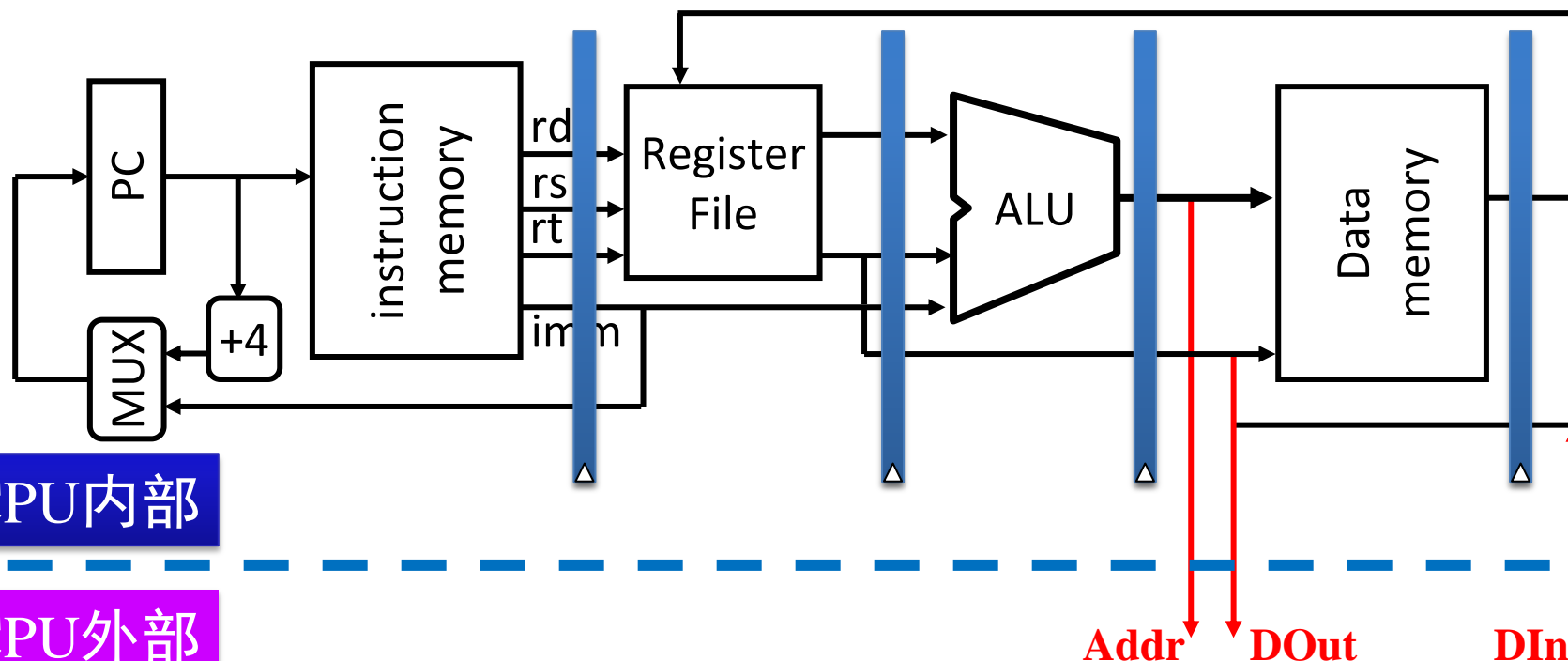
CPU内部

CPU外部

数据通路：增加信号

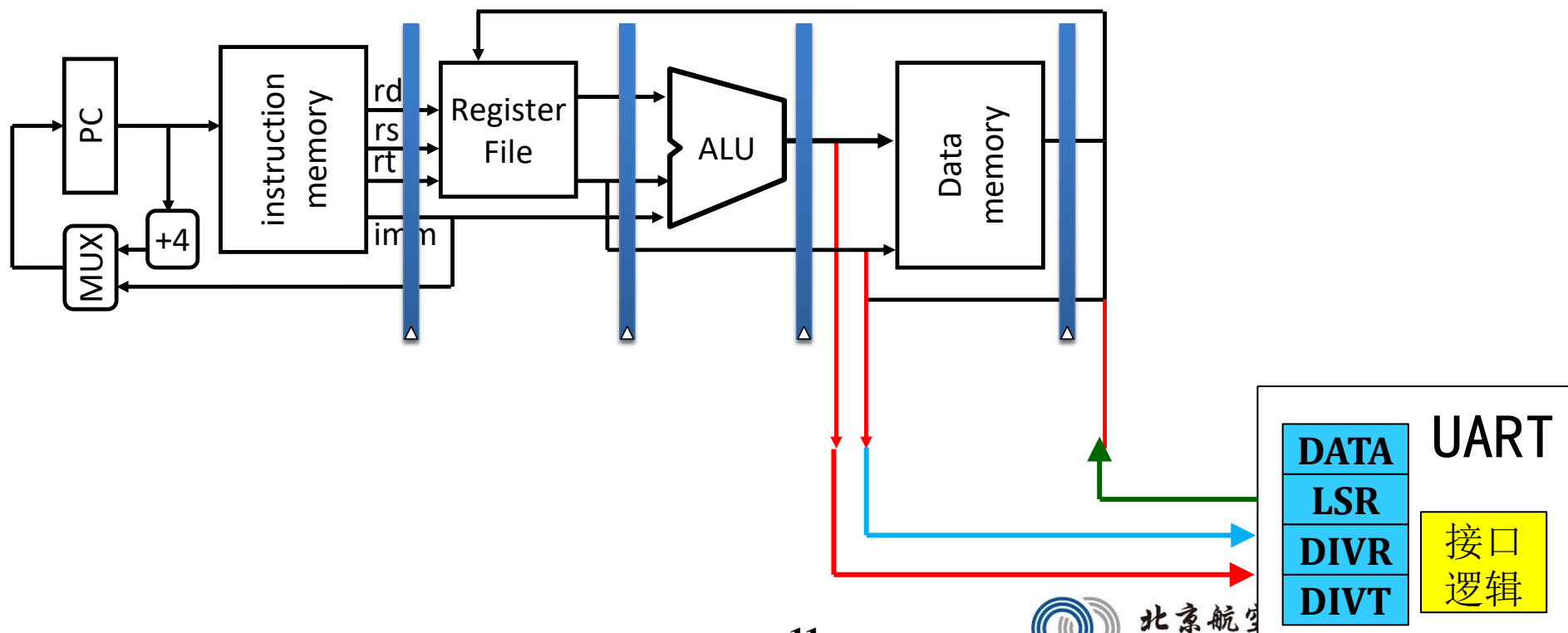
增加必要的地址、数据

- ◆ Addr: ALU计算的存储器地址
- ◆ DOut: CPU写数据
- ◆ DIn: CPU读入的数据



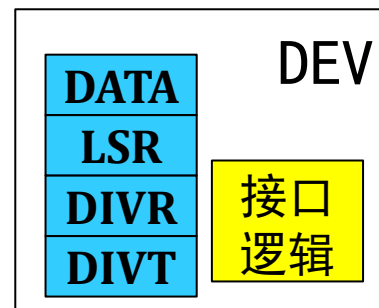
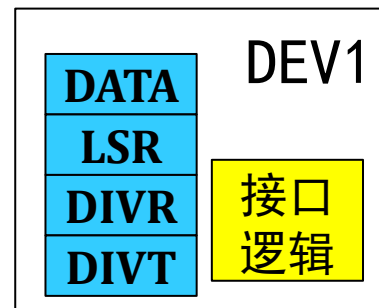
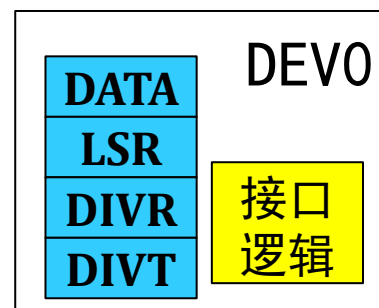
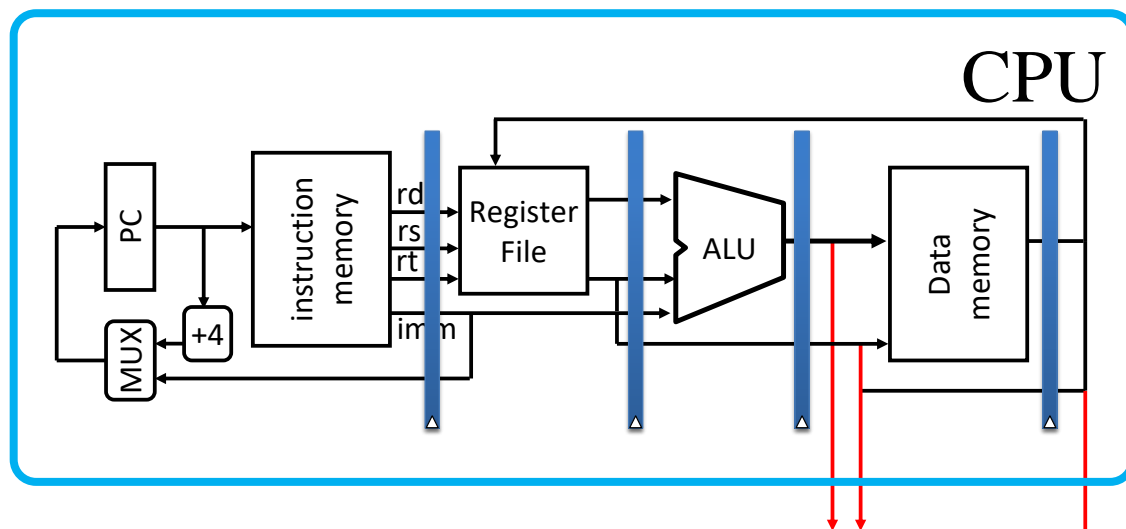
支持对I/O的访问：与设备对接

- 每个设备都有自己的 addr_{dev} 、 din 、 dout
 - Addr_{dev} ：选择设备内部的寄存器。其本质是offset
 - 设备内部的寄存器数量少，因此 Addr_{dev} 位数少
- Q: Addr_{CPU} 位数多，怎么处理？
 - A: 只保留必要的低位地址即可



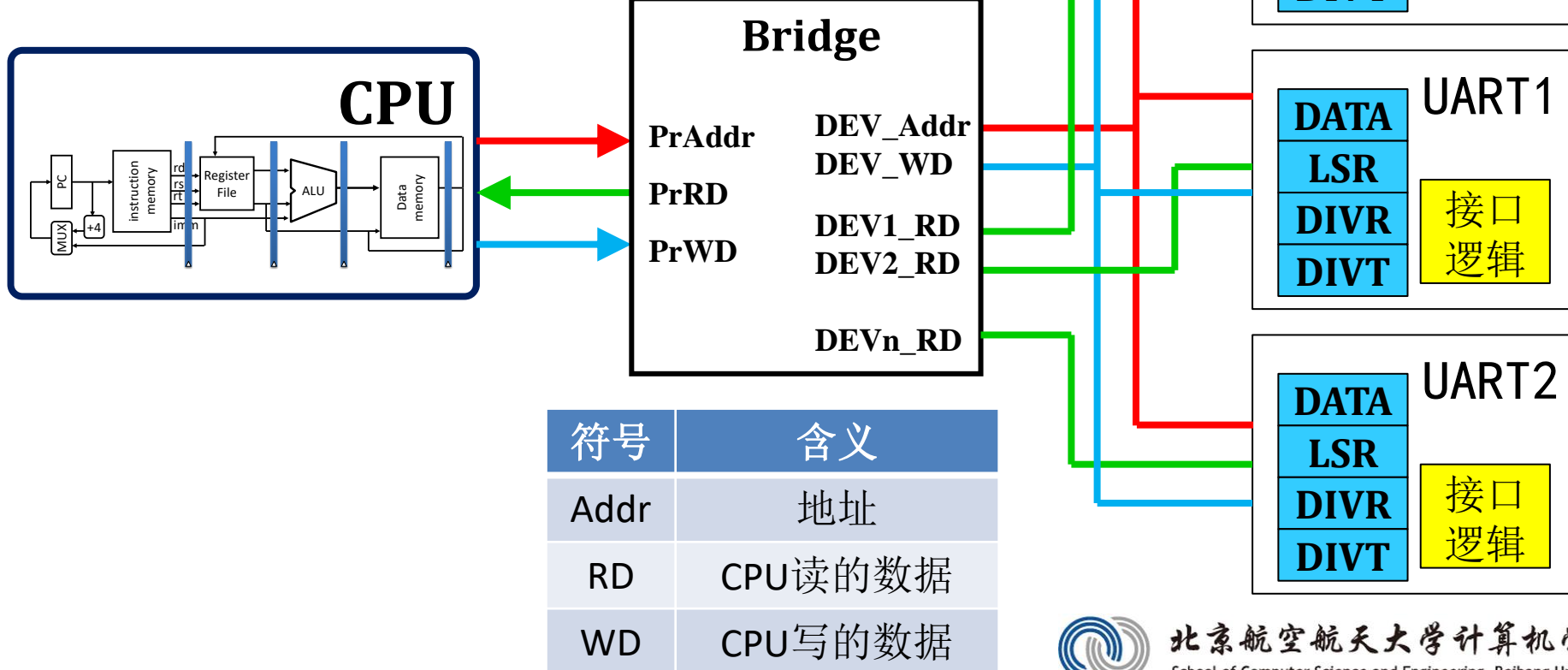
多个设备怎么办？

- CPU不能为每个设备都提供一套地址/数据
 - 否则会导致CPU设计变得复杂



增加新模块：Bridge

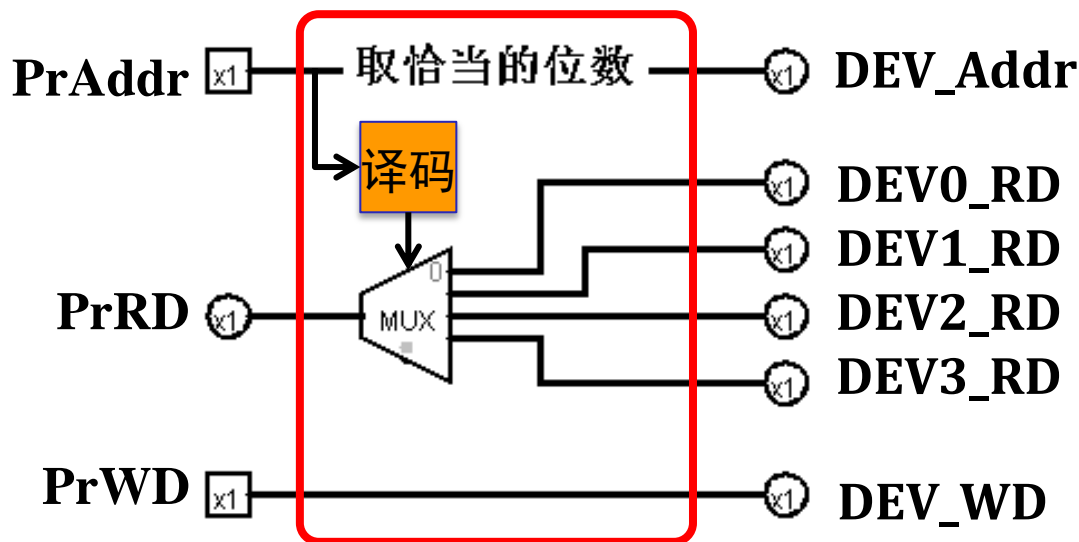
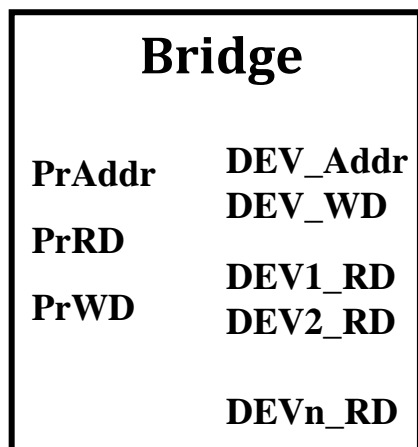
- Bridge：类似与网络switch
 - CPU侧：1组接口。设备侧：N组接口
- 1组地址/写数据，N组读数据
 - CPU读：数据汇聚
 - CPU写：数据派发



Bridge功能及内部结构

- 完成地址、数据转换，控制信号的产生

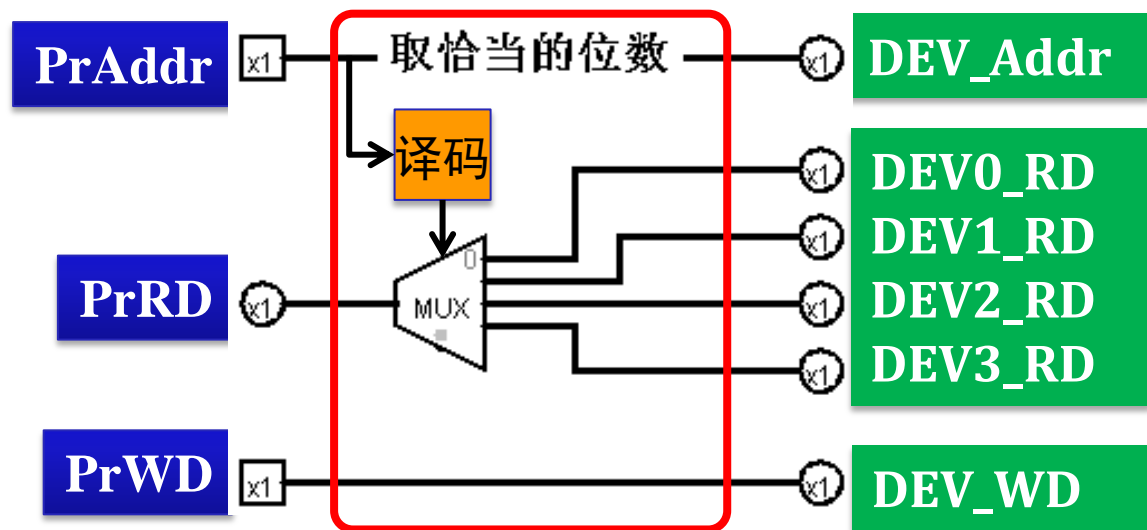
- 地址
- 读数据
- 写数据



地址图

- 地址图：所有设备在地址空间的分布区域
 - ▣ CPU读写设备(其实是程序员)必须知道设备地址
 - ▣ Bridge也必须知道设备，否则无法完成译码
 - ▣ 示例：假设设备0~3均需要256B的地址空间需求

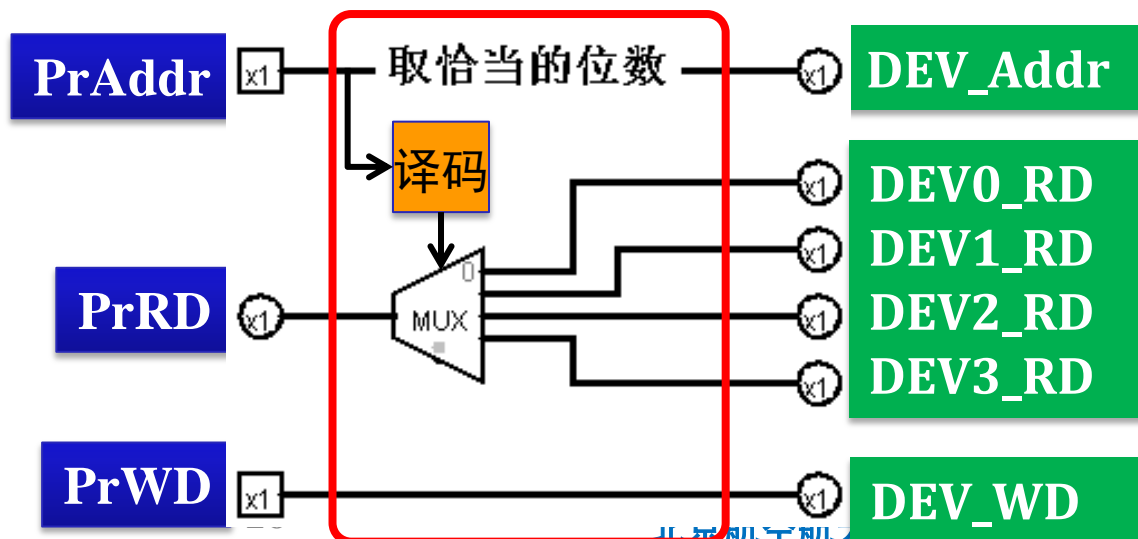
设备	MIPS地址范围	占用空间
DEV0	A0000000 _H ~ A00000FF _H	256字节
DEV1	A0000100 _H ~ A00001FF _H	256字节
DEV2	A0000200 _H ~ A00002FF _H	256字节
DEV3	A0000300 _H ~ A00003FF _H	256字节



Bridge功能(1): 输出地址^{1/2}

- DEV_Addr地址: 将PrAddr[X:2]直接输出即可
 - ▣ X: 由N个设备中地址空间需求最大者决定
 - ▣ 所有设备都只接入各自必要的地址
- 示例: 由于DEV3的地址空间为1MB, 因此X为19

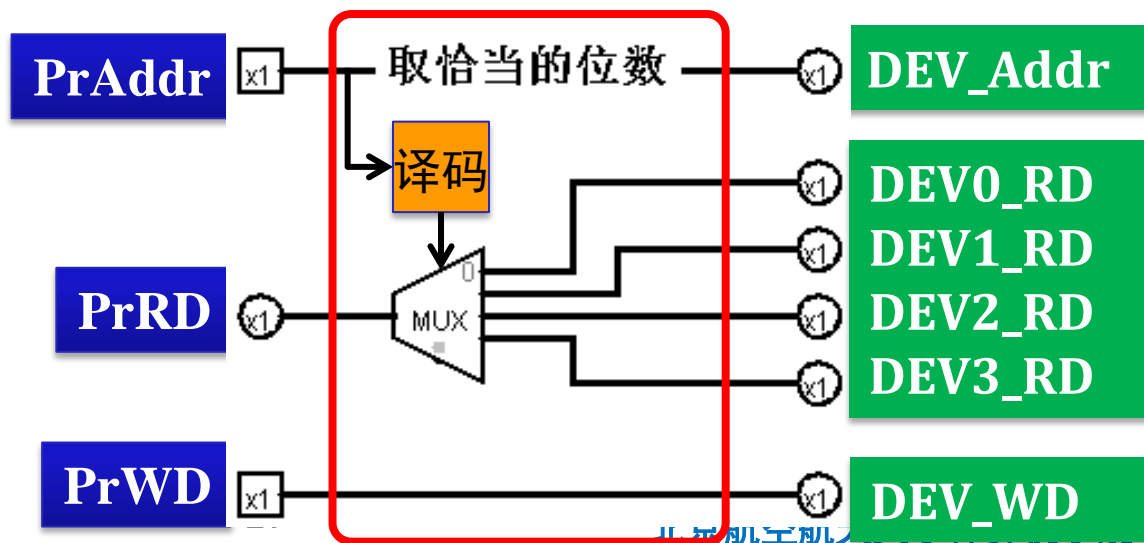
设备	MIPS地址范围	占用空间
DEV0	A0000000 _H ~ A00000FF _H	256字节
DEV1	A0000100 _H ~ A00001FF _H	256字节
DEV2	A0000200 _H ~ A00002FF _H	256字节
DEV3	A0100000 _H ~ A01FFFFF _H	1MB字节



Bridge功能(1): 输出地址^{2/2}

- DEV0~2: 引入DEV_Addr[7:2]即可
 - ▣ DEV0~2地址空间需求: 256B
- DEV3: 必须引入DEV_Addr[19:2]
 - ▣ DEV3地址空间需求: 1MB

设备	MIPS地址范围	占用空间
DEV0	A0000000 _H ~ A00000FF _H	256字节
DEV1	A0000100 _H ~ A00001FF _H	256字节
DEV2	A0000200 _H ~ A00002FF _H	256字节
DEV3	A0100000 _H ~ A01FFFFFF _H	1MB字节



Bridge功能(2): 地址匹配

■ 设备地址译码

□ 设备基地址: 分为高位和低位

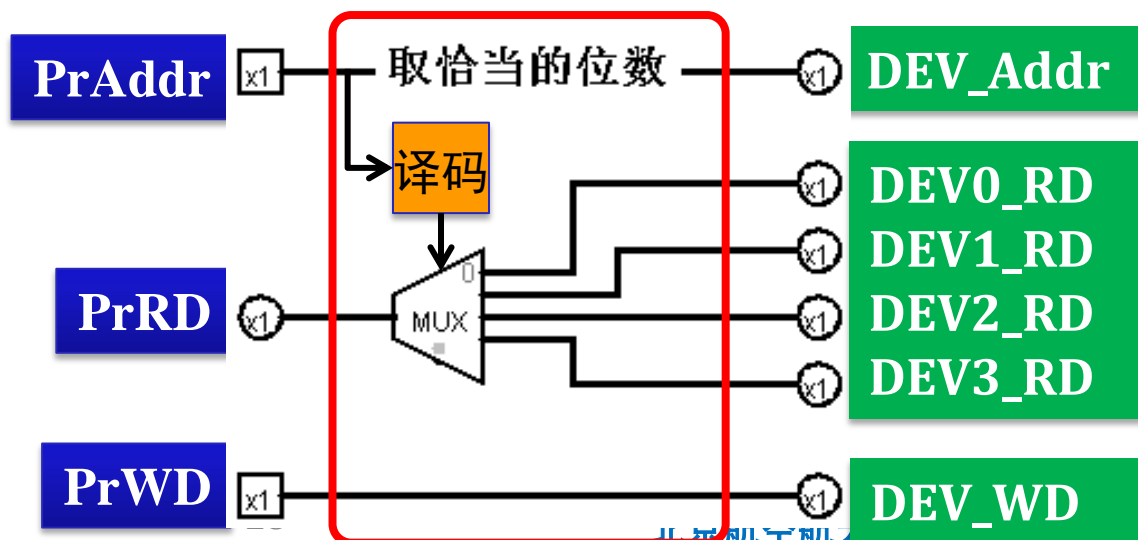
- ◆ 基地址低位: 位数由设备占用空间大小决定, 也就是偏移地址的位数
- ◆ 基地址高位: Bridge用于译码选择

31	X+1	X	0
基地址高位		基地址低位	

设备0 **A0000000**_H
设备1 **A0000100**_H
设备2 **A0000200**_H
设备3 **A0100000**_H

设备

设备	MIPS地址范围	占用空间
DEV0	A0000000 _H ~ A00000FF _H	256字节
DEV1	A0000100 _H ~ A00001FF _H	256字节
DEV2	A0000200 _H ~ A00002FF _H	256字节
DEV3	A0100000 _H ~ A01FFFFF _H	1MB字节



Bridge功能(2): 地址匹配

■ 设备地址译码

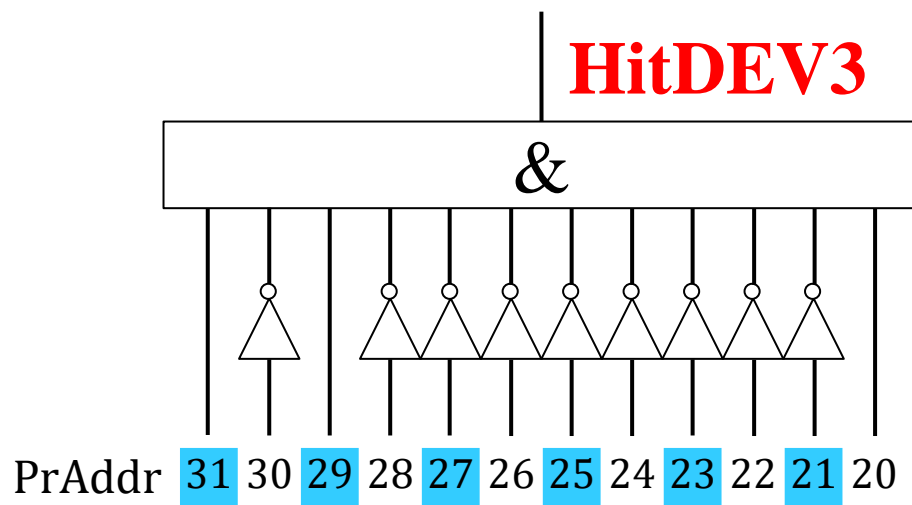
- 为每个设备产生一个译码信号

设备0 **A0000000**_H
设备1 **A0000100**_H
设备2 **A0000200**_H
设备3 **A0100000**_H

Verilog
样例

```
assign HitDEV0 = (PrAddr[31:8] == 'hA00000) ;  
...  
assign HitDEV3 = (PrAddr[31:20] == 'hA01) ;
```

HitDEV3
电路模型



Bridge功能(3): CPU读数据

- 所有设备的**数据输出**汇聚至CPU的**数据输入**
- MUX的控制由PrAddr中某些位译码决定

常规
写法

```
assign PrRD = (HitDEV0) ? DEV0_RD :  
              (HitDEV1) ? DEV1_RD :  
              . . .  
              DEV3_RD ;
```

支持
Debug
写法

```
assign PrRD = (HitDEV0) ? DEV0_RD :  
              (HitDEV1) ? DEV1_RD :  
              . . .  
              (HitDEV3) ? DEV3_RD :  
              `DEBUG_DEV_DATA ;
```

_Addr

0_RD

1_RD

2_RD

3_RD

PrWD

x1

x1

DEV_WD

Bridge功能(4): CPU写数据

- CPU写数据: 连接至所有设备的输入

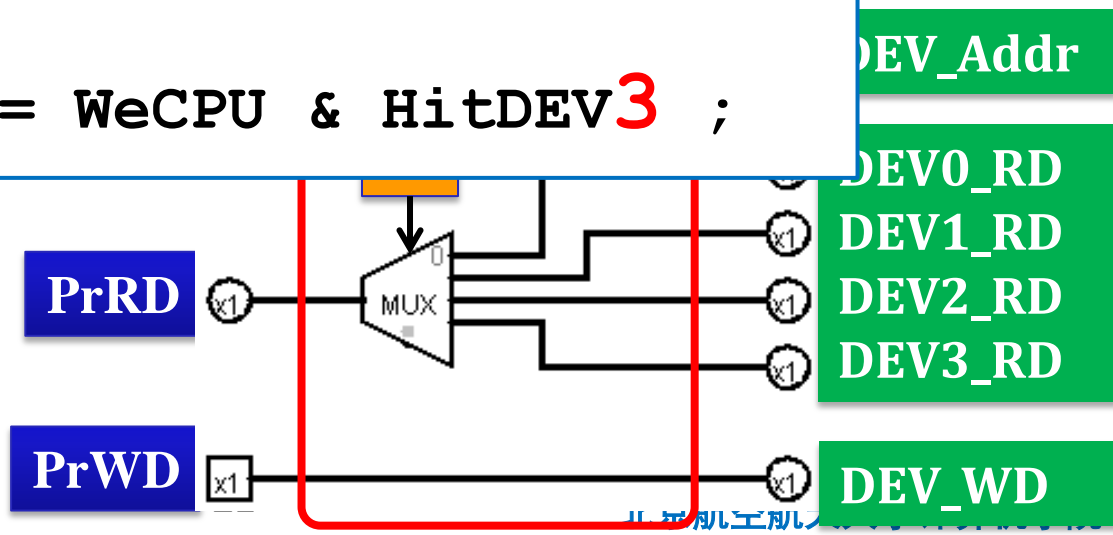
- 直通输出, 不需要再转换

- 控制信号: We

- 有多少个设备, 就需要多少个We

- Verilog样例代码:

```
assign WeDEV3 = WeCPU & HitDEV3 ;
```



问题：谁来区分存储器和设备？

- 假设\$*s0*为0xA000_0000
- `lw $t0, 0($s0)`：是否同时会读存储器和设备？

设备	MIPS地址范围
DEV0	A0000000 _H ~ A00000FF _H
DEV1	A0000100 _H ~ A00001FF _H
DEV2	A0000200 _H ~ A00002FF _H
DEV3	A0000300 _H ~ A00003FF _H

